Jedną z bardziej spektakularnych cech najnowszych komputerów osobistych są ich możliwości graficzne. Otwierają się nowe perspektywy nie tylko przed fanatykami gier komputerowych, ale i przed projektantami najrozmaitszych urządzeń. Do poważniejszych zastosowań niezbędne są jednak odpowiednie urządzenia we-wy i oprogramowanie. Spadające ceny sprzętu świadczą, że grafika komputerowa niebawem znajdzie się w zasięgu możliwości co bardziej zdeterminowanych amatorów mikroinformatyki. Dlatego z pewnym wyprzedzeniem spróbujemy przekazać Czytelnikom pakiet odpowiednich informacji. Rozpoczynamy od skrótowego opisu normy GKS, który przyda się do zrozumienia następnych tekstów na ten temat.

Jako ciekawostkę podajemy, że ilustracje do poniższego tekstu zostały wykonane za pomocą plotera podłączonego do ELWRO-513 (również opisy na rysunkach).

Norma GKS

Trudno uwierzyć, że grafika komputerowa ma już ponad trzydzieści lat! Pierwsze monitory graficzne pojawiły się w 1951 roku, natomiast pisaki X-Y — w 1953. Graficzne urządzenia wejściowe stosowane są w świecie od roku 1958 (pióro świetlne). Początek kolorowych monitorów graficznych datuje się na lata 1962—65.

Do niedawna cena zestawu podstawowych graficznych urządzeń zewnętrznych była bardzo wysoka (ok. 400 tys. dol.). Obecnie kwota ta spadła do tego stopnia (kilkadziesiąt razy), że mówi się już o grafice mikrokomputerowej.

W latach siedemdziesiątych prowadzone były intensywne prace nad standardem dla grafiki komputerowej (m.in. GSPC Core w USA, GINO-F w Wielkiej Brytanii, IDIGS w Holandii i Norwegii, GKS w RFN). Dopiero w 1982 roku ISO (International Organization for Standardization) uznała **GKS** (Graphical Kernel System — system jądra graficznego) za normę międzynarodową.

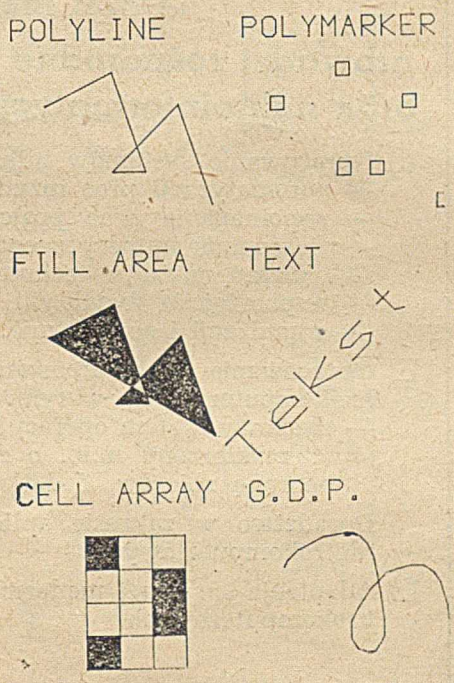
Obecnie wszystkie liczące się w świecie firmy komputerowe stosują się do tej normy, ponieważ zgodność z GKS jest niezbędna dla sukcesu w sprzedaży sprzętu i oprogramowania graficznego.

Szczegółowy opis normy znaleźć można w [1]. Informacje na temat powstania GKS zaczerpnąć można z [2]. Godną polecenia jest też książka [3].

Rys. 1. Miejsce GKS w programie użytkowym

Do graficznych urządzeń wyjścia zaliczone są: monitory graficzne monochromatyczne i kolorowe, pisaki X-Y oraz drukarki graficzne. Przedmiotem normy GKS są zasady tworzenia i przetwarzania obrazów. Dla GKS obrazy składają się ze skończonej ilości podstawowych bloków, tzw. prymitywów, do których zaliczono:

— łamaną (ang. **POLYLINE**), czyli ciąg odcinków o wspólnych końcach
— zbiór punktów (ang. **POLYMARKER**) znaczonych pewnymi symbolami graficznymi
— obszar ograniczony wielokątem (ang. **FILL AREA**), zakreskowany według pewnej zasady lub wypełniony ustalonym kolorem
— tekst alfanumeryczny (ang. **TEXT**) o ustalonych parametrach
— obraz dwuwymiarowej macierzy (ang. **CELL ARRAY**), której elementami są kody kolorów punktów



Rys. 2. Prymitywy reprezentacji GKS

— łuki krzywych gładkich (ang. **GENERALIZED DRAWING PRIMITIVE**), takich jak okrąg, elipsa czy parabola.

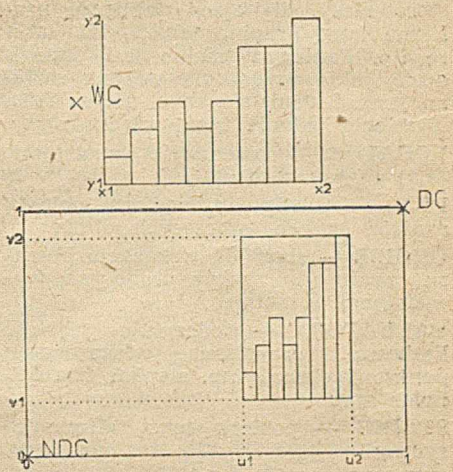
Z każdym prymitywem skojarzone są pewne parametry (atrybuty) określające np.:

— rodzaj linii (ciągła, przerywana, kropkowana lub inna)
— znak graficzny do obrazowania punktów (kropka, kółko, krzyżyk, gwiazdka lub inny)
— sposób wypełniania obszarów (nachylenie i odstęp kresek, kolor lub inny)
— wysokość liter, kierunek pisania tekstu, punkt odniesienia, krój liter, alfabet, itp.
— kolor linii, znaków, obszarów, liter oraz tła.

GKS wyodrębnia trzy układy współrzędnych:

— układ współrzędnych rysunku — WC (ang. world coordinates)
— znormalizowany układ współrzędnych urządzenia — NDC (ang. normalized device coordinates)
— układ urządzenia graficznego — DC (ang. device coordinates).

Znaczenie poszczególnych układów współrzędnych ilustruje rysunek 3.

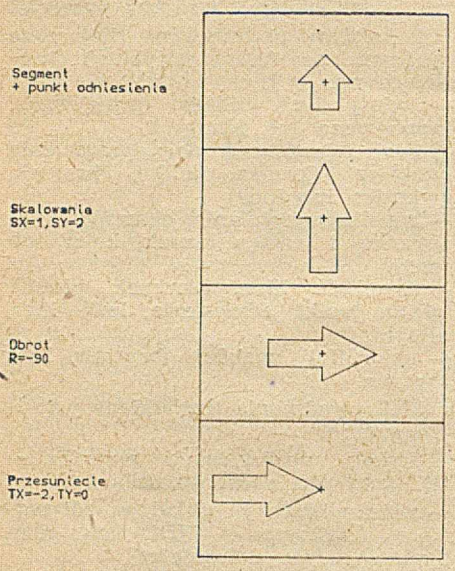


Rys. 3. Układy współrzędnych

Przyjmijmy, że obraz jest pewnym zdaniem sformułowanym w języku o gramatyce atrybutowej. Segmentami nazwiemy „wyrazy", które występują w tym zdaniu. Poprawne wyodrębnienie segmentów ułatwia przetwarzanie obrazu (animację), czyli zmianę atrybutów przyporządkowanych segmentom. Do atrybutów należą:

— punkt odniesienia segmentu
— wektor translacji
— kąt obrotu wokół punktu odniesienia
— skala jednokładności o środku w punkcie odniesienia
— znacznik „obcinania" (ang. clipping)

— znacznik widzialności (znikanie i pojawianie się)
— wskaźnik priorytetu (przesłanianie segmentów o niższym priorytecie)
— znacznik intensywnego świecenia (ang. highlighting).



Rys. 4. Transformacje segmentów

Wśród graficznych urządzeń wejścia należy wymienić: pióro świetlne, monitor „dotykowy", myszkę, drążek sterowniczy, digitajzer, tabliczkę graficzną (ang. graphic tablet), klawiaturę itp. Ponieważ trudno jest przewidzieć, jakie urządzenia wejścia zostaną jeszcze skonstruowane, norma GKS wymienia jedynie typy urządzeń:

LOCATOR — przekazuje współrzędne (x,y) punktu obrazu

PICK — służy do wyboru obiektu spośród innych tworzących obraz

CHOISE — służy do wskazania obszaru obrazu (np. wybór propozycji wymienionej w menu)

VALUATOR — odczytuje wartość liczbową przypisaną wskazanemu punktowi obrazu (np. stopień szarości, kolor itp.)

STRING — pobiera tekst alfanumeryczny

STROKE — przesyła ciąg punktów wyznaczających kontur.

Oprócz podziału urządzeń na typy, GKS uwzględnia trzy rodzaje interakcji:

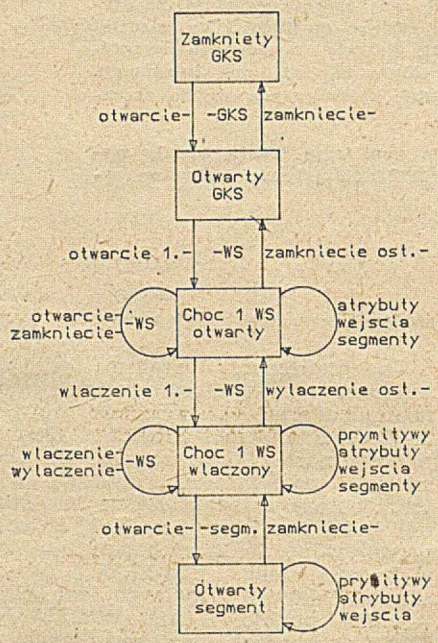
REQUEST — urządzenie wejścia odpowiada na żądanie nadawania, wysłane z programu użytkowego

SAMPLE — program użytkowy i proces wejścia przebiegają równolegle, przy czym rolę nadrzędną pełni program użytkowy

EVENT — program użytkowy i proces wejścia przebiegają równolegle, przy czym rolę nadrzędną pełni proces wejścia.

Dla uniknięcia powtarzania długotrwałego procesu tworzenia obrazu, gotowe rysunki (lub ich fragmenty) można przechowywać w pamięci masowej. Norma GKS nie narzuca sposobu kodowania obrazu, określa jedynie czynności zapisywania i odczytywania.

Do programowania tych czynności przewidziane są funkcje:
— zapisu obrazu w zbiorze; parametrami tej funkcji są: numer logiczny urządzenia graficznego, sposób kodowania obrazu w zbiorze oraz długość i numer rekordu
— rozpoznawania typu danych przed odczytem
— odczyt danych graficznych ze zbioru pamięci masowej
— interpretacja (rozkodowanie) danych i generacja obrazu.



Rys. 5. Graf stanów GKS

Obsługa urządzeń graficznych przebiega według schematu przedstawionego na rysunku 5. Jakiekolwiek odstępstwo od tego schematu lub inne błędy semantyczne sygnalizowane są przez diagnostykę błędów. Komunikat o błędzie zapisywany jest w zbiorze pamięci masowej o nazwie zadeklarowanej w trakcie otwierania GKS.

ANTONI URBAN
Computer Studio Kajkowski

LITERATURA

[1] DIN-66252: Informationsverarbeitung-Graphisches Kernsystem (GKS). Funktionale Beschreibung, Alleinverkauf der Normen durch Beuth Verlag GmbH, Berlin, 1983
[2] Encarnaçao J., Strasser W.: Geräteunabhängige graphische System. R. Oldenbourg Verlag München, Wien, 1981
[3] Hopgood F. R. A., Duce D. A., Gallop J. R., Sutcliffe D. C.: Introduction to the Graphical Kernal System (GKS). Academic Press, London, 1983.

Wojewódzki Klub Techniki i Racjonalizacji
Zakład Usług Technicznych
20-919 Lublin, ul. Marii Curie-Skłodowskiej 3
tel. 213-39, 284-64, 55-80-01 w. 226

prowadzi różnorodne usługi techniczne oraz mikroinformatyczne:

☐ Opracowanie systemów mikroinformatycznych na potrzeby:
— automatyzacji prac inżynierskich
— wspomagania prac projektowych i konstrukcyjnych
— sterowania numerycznego
— obliczeń naukowo-technicznych
— wspomagania dydaktyki
— organizacji i zarządzania.

☐ Opracowanie oprogramowania podstawowego systemów mikrokomputerowych, w tym:
— dyskowy system operacyjny SOMIK kompatibilny z CP/M 2.2,rozszerzony m.in. o zegar, znakowanie zbiorów, datę, ochronę zbiorów, dodatkowe możliwości pracy wsadowej.

☐ Doradztwo w zakresie aplikacji i wykorzystania systemów mikrokomputerowych.

☐ Adaptacja sprzętu i modernizacja architektury systemów mikrokomputerowych.

EO/140/K/85

Pomysł przystawki pozwalającej wykorzystać ZX81 jako woltomierz lub generator funkcji wydaje się być wyjątkowo trafny. Oczywiście, nie należy stawiać sprawy na głowie i kupować ZX81 zamiast woltomierza cyfrowego... Jeżeli jednak znalazł się on już w laboratorium — opisywane rozszerzenie jego możliwości może okazać się użyteczne. Pomysł jest dokładnie na miarę możliwości ZX81 (ideę można łatwo przenieść na inny mikrokomputer, np. ZX SPECTRUM) i to go odróżnia od niektórych szaleńczych aplikacji, jak choćby zastosowanie w intensywnym nadzorze medycznym, od przebiegu którego może zależeć życie pacjenta.

Przystawka do ZX81

Opisana przystawka pozwala na wykorzystanie ZX81 jako generatora funkcji, miernika częstotliwości lub programowalnego woltomierza. Zamieszczane programy zostały przetestowane pod kątem typowych zastosowań. Tłumacząc je na język maszynowy można zwiększyć szybkość ich realizacji.

WOLTOMIERZ CYFROWY

Konwerter analogowo-cyfrowy (ang. **ADC**) zrealizowano z wykorzystaniem przetwornika cyfrowo-analogowego (ang. **DAC**) oraz analogowego komparatora poziomów. Program — wysyłając kolejne liczby do DAC — wytwarza na jego wyjściu coraz wyższe poziomy napięciowe, które są porównywane z mierzonym napięciem przez komparator. Zgodność napięć powoduje zmianę stanu logicznego na wyjściu komputera — co jest wykrywane przez program i powoduje jego zatrzymanie.

GENERATOR FUNKCJI

Układ pozwala generować z kwarcową stabilnością falę prostokątną, ciągi pojedynczych impulsów oraz — z pewnymi ograniczeniami — napięcie o dowolnej zależności czasowej.

Żądane przebiegi uzyskuje się za pomocą programu przesyłającego do DAC tablicę liczb odpowiadających napięciom, jakie chcemy uzyskać w kolejnych chwilach. Przybliżamy w ten sposób zadany przebieg funkcją schodkową o zbliżonym kształcie. Zamykając program w pętlę, uzyskujemy przebieg okresowy.

Generację fali prostokątnej można zrealizować znacznie prościej. Uzyskujemy ją stosując pętlę programową napisaną w języku BASIC lub w języku maszynowym. Program sprowadza się do wysyłania do portu we-wy na przemian zer i jedynek. Pętla napisana w języku BASIC daje maksymalną częstotliwość 190 Hz, podczas gdy pętla zapisana w języku maszynowym zapewnia obsługę częstotliwości do 12,5 kHz. Zmianę częstotliwości otrzymujemy wprowadzając w pętlę odpowiednie opóźnienie.

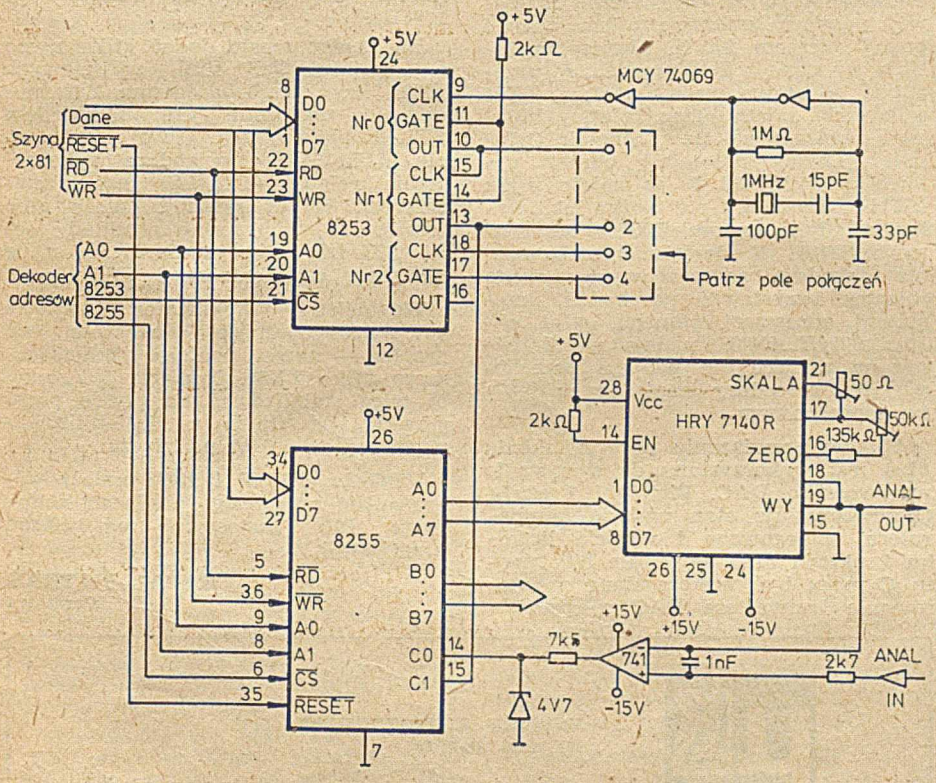
```
  4 REM POMIAR NAPIECIA
 10 POKE 8195,137
 20 FOR D=0 TO 255
 30 POKE 8192,D
 40 LET A=PEEK 8194
 50 IF A=2*INT (A/2) THEN GOTO 70
 60 NEXT D
 70 PRINT D;"V"
 80 GOTO 20
```

```
  4 REM GENERATOR PRZEBIEGU
  6 REM PROSTOKATNEGO
 10 POKE 8195,137
 20 INPUT F
 30 CLS
 40 LET N=1000000/F
 50 LET M=INT (N/256)
 60 LET L=INT ((N/256-M)*256)
 70 POKE 8199,54
 80 POKE 8196,L
 90 POKE 8196,M
100 PRINT 1000000/INT N;" HERTZ"
110 GOTO 20
```

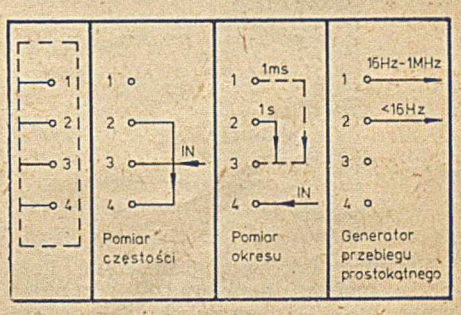
```
  1 REM 1234567890123456
  4 REM GENERACJA FALI SINUSOI-
DALNEJ
 20 FOR N=1 TO 16
 30 LET P=16513+N
 40 LET S=INT (128*(1+COS (2*PI
*N/16)))
 50 POKE P,S
 60 NEXT N
 70 FOR N=1 TO 16
 80 LET P=16513+N
 90 LET Q=PEEK P
100 POKE 8192,Q
110 NEXT N
120 GOTO 70
```

Asymetria opóźnień przełączania dla 0 i 1 daje efekt generowania impulsów dodatnich lub ujemnych. Zmiana opóźnienia za każdym obiegiem pętli pozwala na „przemiatanie" częstotliwości.

Falę prostokątną można jeszcze prościej otrzymać — korzystając ze sprzętowych udogodnień, jakich dostarcza układ 8253 zawierający trzy niezależne 16-bitowe liczniki programowane. Każdy z nich może pracować w jednym z sześciu trybów, wśród których znajdują się: tryb generacji impulsów oraz tryb generacji fali prostokątnej. Wystarczy przesłać słowo kontrolne i dwa bajty informacji do każdego z liczników, aby te — w zależności od zaprogramowanego trybu pracy — przejęły zadanie generacji impulsów czy też przebiegu prostokątnego umożliwiając ZX81 zajęcie się innymi problemami. Przebiegi wyjściowe układu 8253 powstają w wyniku podziału częstotliwości wejściowej (mogącej sięgać 2 MHz) przez



Rys. 1. Schemat ideowy przystawki



Rys. 2. Pole połączeń

zaprogramowaną wartość. Stopień podziału musi być większy od 1 a fala prostokątna przy nieparzystym współczynniku podziału ma poziom wysoki, o jeden okres zegara dłuższy od poziomu niskiego.

CZĘSTOTLIWOŚCIOMIERZ

Ta funkcja przystawki jest zrealizowana całkowicie w oparciu o własności układu 8253. Mierzona częstotliwość podawana jest na wejście licznika 2, bramkowanego impulsami o długości 1 s powstałymi z podzielenia zegara wzorcowego 1 MHz w licznikach 0 i 1 połączonych szeregowo. W tej konfiguracji liczniki 0 i 1 pracują w trybie 3, natomiast licznik 2 — w trybie 0.

Tabela 1. Format słowa kontrolnego dla 8253

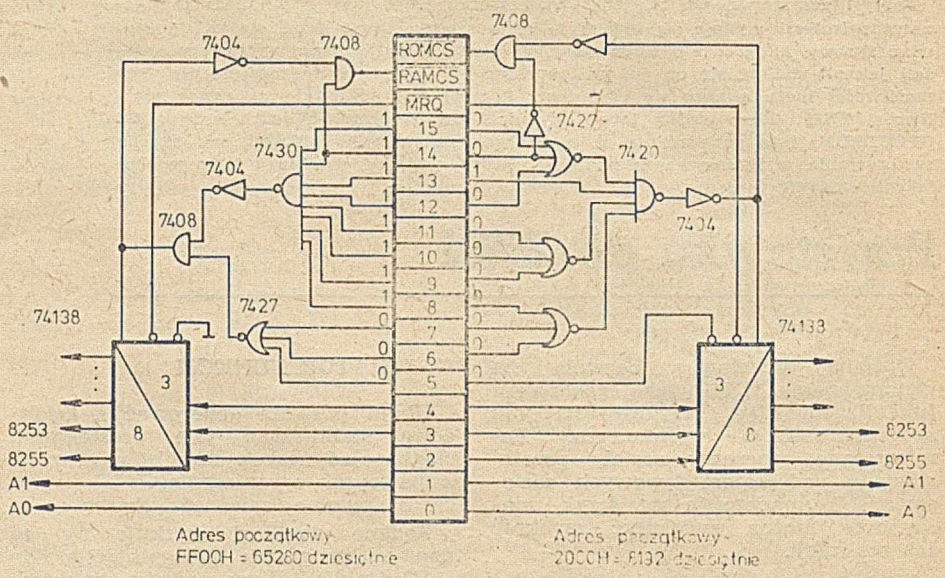
Bit	Znaczenie	Nr 0	Nr 1	Nr 5
7	nr licznika	0	0	1
6		0	1	0
5	tryb dostępu	1	1	1
4		1	1	1
3	tryb pracy	1	1	0
2		1	1	0
1		1	1	0
0	dwójkowo BCD	0	0	0
Do wpisania		54	118	176

abela 2. Format słowa kontrolnego dla 8255

Bit	Znaczenie	Dane
7	tryb/zapis C	1
6	tryb pracy	0
5		0
4	1-linie A wejście	0
3	1-linie C4÷7 wejście	1
2	tryb pracy B	0
1	1-linie B wejście	0
0	1-linie C0÷3 we'ście	1
Do wpisania		137

```
  4 REM POMIAR CZESTOSCI
 10 POKE 8195,137
 20 POKE 8199,54
 30 POKE 8196,232
 40 POKE 8196,3
 50 POKE 8199,118
 60 POKE 8197,208
 70 POKE 8197,7
 80 POKE 8199,176
 90 POKE 8198,225
100 POKE 8198,255
110 LET A=(PEEK 8194)/2
120 IF A=2*INT (A/2) THEN GOTO
110
130 LET A=(PEEK 8194)/2
140 IF NOT (A=2*INT (A/2)) THEN
GOTO 130
150 POKE 8199,128
160 LET X=PEEK 8198
170 LET Y=PEEK 8198
180 CLS
190 PRINT 65536-Y*256+X;" HERTZ
"
200 GOTO 80
```

Stosując tak zaprogramowane liczniki można doprowadzić impulsy wzorcowe do wejścia zliczającego licznika 2. Uzyskujemy w ten sposób możliwość pomiaru czasu trwania impulsu doprowadzonego na wejście bramkujące licznika 2. Ponieważ w układzie 8253 liczniki liczą „wstecz", należy pamiętać o konieczności wpisania (przed pomiarem okresu) liczby większej od spodziewanego czasu trwania impulsu, do licznika 2. Różnica między stanem końcowym a początkowym tego licznika da długość mierzonego impulsu.



Rys. 3. Dekoder adresów

KONSTRUKCJA PRZYSTAWKI

Wszystkie rejestry przystawki umieszczone są w przestrzeni adresowej pamięci, zajmując osiem pozycji ponad adresem 2000H. Miejsce to zostało wybrane z tego powodu, że przestrzeń adresowa we-wy jest w ZX81 całkowicie zajęta przez rejestry układu zarządzającego (tzw. Sinclair Computer Logic), natomiast adresy pamięci powyżej 32 K są zajęte przez MEMOPAK-64 K i nie można ich „wyłączyć". Posiadaczom ZX'ów bez rozszerzenia 64 K doradzam zastosowanie dekodera adresów wybierającego rejestry przystawki na końcu przestrzeni adresowej pamięci, gdyż obszar 2000H...4000H nadaje się świetnie na miejsce dla pamięci przechowującej procedury napisane w języku maszynowym.

W tym rozwiązaniu port A układu 8255 jest wykorzystany do sterowania przetwornika DAC. Bity 0 i 1 portu C odwzorowują stan wyjścia komparatora analogowego i wyjścia licznika 1 układu 8253. Pozostałe bity portu C mogą być wykorzystane do sterowania konwertera DAC o rozdzielczości większej niż 8 bitów. Port B jest całkowicie niewykorzystany i może być użyty do innych celów.

Pomiary częstotliwości i okresów dokonywane za pomocą tej przystawki będą tym dokładniejsze, im bardziej precyzyjny będzie zegar wzorcowy sterujący układem 8253. ZX81 jest wyposażony w generator kwarcowy, jednak jego częstotliwość (3,75 MHz) nie jest zbyt wygodna do celów pomiarowych. Z tego powodu przystawka została wyposażona w prosty generator zrealizowany na inwerterze CMOS z kwarcowym rezonatorem 1 MHz.

Na koniec warto dodać kilka uwag o obciążalności linii wyjściowych ZX81. Szyny mikroprocesora Z80 mogą być obciążane jednym układem TTL i pojemnością 130 pF. Dołączenie MEMOPAK'a powoduje wykorzystanie połowy (TTL-LS) jego obciążalności. W tej sytuacji należy albo zastosować w przystawce układy serii TTL-LS, albo zbuforować szynę adresową wzmacniaczami TTL.

Wewnętrzny stabilizator napięcia 5 V mikrokomputera znacznie się grzeje — nawet podczas obsługi samego tylko ZX81. Napięcie 5 V warto więc stabilizować na płytce przystawki, korzystając z niestabilizowanego napięcia 9 V na złączu.

Napięcia +15 V i 15 V należy doprowadzić z osobnego zasilacza sieciowego lub zastosować odpowiedni przetwornik.

RAFAŁ PIETRAK
Warszawa

μK
prowadzi
ANDRZEJ J. PIOTROWSKI
tel. dom. 48-22-85

Mikrokomputer MERITUM-I

Produkcję MERITUM-I podjęły Zakłady Urządzeń Komputerowych MERA-ELZAB w Zabrzu. MERITUM-I zostało opracowane [4] po przeprowadzeniu analizy wielu popularnych w krajach zachodnich komputerów osobistych oraz po uwzględnieniu dostępności w naszych warunkach niezbędnych podzespołów elektronicznych. MERITUM-I zbliżone jest od strony sprzętowej do TRS-80/II firmy TANDY — RADIO-SHACK, natomiast oprogramowanie jest podobne do jednej z wersji języka BASIC opracowanej przez firmę MICROSOFT.

MERITUM-I nie jest systemem modularnym, lecz wytwórca przewiduje możliwość jego rozbudowy. Adresowany jest do szerokiego kręgu odbiorców jako prosty mikrokomputer o wszechstronnym zastosowaniu.

MERITUM-I został zbudowany z podzespołów elektronicznych produkowanych w krajach RWPG. Zastosowano w nim mikroprocesor U880D (NRD-owski odpowiednik Z-80). Pamięć operacyjna RAM składa się z 16 KB zbudowanych z radzieckich pamięci dynamicznych 16K x 1 typu K 565 RU3 oraz z 1 KB pamięci statycznej (odpowiednik 2114). Oprogramowanie podstawowe może zajmować do 14 KB ROM, z czego 12 KB wykorzystuje rezydujący interpreter języka BASIC. Oprogramowanie fabryczne zawarto w pamięciach EPROM typu K573 RF2. Pamięć obrazu (video-RAM) zajmuje 1 KB.

Funkcje wejścia-wyjścia realizowane są z wykorzystaniem krajowych odpowiedników układów z rodziny 8080. Sprzęg szeregowy odpowiada standardowi RS 232C i możliwościom układu 8251. Zaleca się wykorzystanie go do sterowania drukarką (np. typu D-100). Sprzęg równoległy zrealizowano na MCY 7855 (odpowiednik 8255) — 22 bity są dostępne dla użytkownika. Sprzętowa realizacja zależności czasowych jest możliwa dzięki zastosowaniu układu zliczającego odpowiadającego 8253.

Zewnętrzną pamięcią masową jest standardowy magnetofon kasetowy. Częstotliwość sygnału zapisu wynosi 1 kHz, lecz efektywna prędkość transmisji wynosi tylko 500 bodów. W praktyce zapis na taśmę całej pojemności pamięci RAM trwa prawie 5 minut. Przyjęty sposób zapisu nie zawiera nadmiarowości kodu, a więc samoczynna korekta błędu nie jest możliwa. Ryzyko posługiwania się błędnym zapisem lub odczytem zmniejsza zastosowanie sumy kontrolnej oraz możliwość programowego sprawdzenia zapisu.

Mikrokomputer MERITUM-I generuje zespolony sygnał wizyjny, co umożliwia przyłączenie monitora telewizyjnego (np. typu NEPTUN 156 lub MTU 3159). Przyłączenie telewizora jest możliwe jedynie za pośrednictwem dodatkowego adaptera (modulator pracujący w kanale 1-6). Ekran jest zorganizowany w 16 wierszy, a liczbę znaków w wierszu można programowo ustawić na 32 lub 64. Oprócz znaków alfanumerycznych można wyświetlać 64 specjalne symbole graficzne. Ekran może być również traktowany jak zbiór dostępnych programowo punktów (128 poziomo i 48 pionowo).

Klawiatura kontaktronowa w układzie QWERTY obsługiwana jest programowo. Uzupełniają ją klawisze specjalne:

RESET: do zerowania całego mikrokomputera, w tym pamięci i portów

NMI: do zerowania mikrokomputera bez zerowania pamięci. Wykorzystywany w przypadku „zawieszenia się" systemu, gdy zależy nam na zachowaniu programu w pamięci RAM

CLEAR: zeruje ekran monitora

ENTER: kończący wpisywanie linii programu, zleceń systemowych itp.

BREAK: przerywający wykonywanie programu.

Układy mikrokomputera umieszczone są w obudowie o wymiarach: 380 × 250 × 90 mm, razem z klawiaturą. Zasilacz sieciowy znajduje się w podobnej (lecz węższej — 120 mm) obudowie i dostarcza napięć: +5 V/3 A, +12 V/1 A oraz —5V/0,2 A. Obudowy wykonano z tworzywa sztucznego.

MERITUM-I posiada (w pamięci EPROM) interpreter języka BASIC. Poniżej opiszemy niektóre cechy charakterystyczne dla zastosowanej wersji tego języka.

Nazwy zmiennych i tablic są rozpoznawane przez dwa pierwsze znaki, lecz mogą być dłuższe. Istnieją cztery typy zmiennych. Mogą one być określone przez definiowanie lub przez dodanie do nazwy odpowiedniego symbolu.

DEFINT (%) — dla zmiennych całkowitych. Zajmują one w pamięci po 2 bajty i mają zakres od —32768 do +32767.

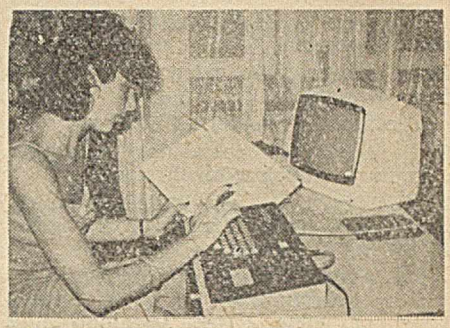
DEFSGN (!) — dla zmiennych o pojedynczej precyzji. Zajmują one w pamięci po 4 bajty i obejmują zakres liczbowy $\pm 1{,}7 \cdot 10^{38}$, dając 6 cyfr znaczących.

DEFDBL (#) — dla zmiennych podwójnej precyzji. Zajmują one w pamięci po 8 bajtów i obejmują zakres liczbowy $\pm 1{,}7 \cdot 10^{38}$, dając 16 cyfr znaczących.

DEFSTR ($) — dla zmiennych łańcuchowych. Mogą one mieć długość do 255 znaków, a każdy znak zajmuje 1 bajt pamięci.

Instrukcje CDBL i CSNG pozwalają na konwersję typu zmiennej między pojedynczą i podwójną precyzją.

Najmniejszą wartością indeksu w tablicy jest zero. Tablice mogą być wielowymiarowe. Brak jest jednak



MERITUM-I z monitorem NEPTUN 156
Fot. Zbigniew Chrześcijański

standardowych działań na nich (np. funkcji macierzowych).
Dostępne są natomiast standardowe operacje na zmiennych łańcuchowych. Są również funktory logiczne AND-OR-NOT.

Brak jest funkcji definiowanych przez użytkownika. W rezultacie nie ma możliwości przekazywania w ogólny sposób. parametrów do procedur Zmusza to do stosowania różnych „chwytów".

Instrukcje sterujące mają wygodną postać: **IF-THEN-ELSE** oraz **ON X GOTO N_1, N_2,...,N_k** i **ON X GOSUB N_1, N_2,...,N_k**.
Nie ma operacji na plikach, co jest konsekwencją wykorzystywania magnetofonu jako pamięci zewnętrznej. Można jednak rozdzielić w kasecie programy i dane tak przy zapisie, jak i przy odczycie.

Użyteczne są instrukcje obsługi błędów:

ON ERROR GOTO — skok do podprogramu z obsługą błędu

RESUME — koniec podprogramu obsługi błędu

ERL — wyznaczenie numeru linii, w której wystąpił błąd

ERR — wyznaczenie kodu zaistniałego błędu

ERROR — symulacja błędu do badania reakcji programu.

Dzięki nim błędy nie muszą przerywać wykonywania programu, a jeśli nie można tego uniknąć, to wznowienie pracy może być szybsze.

Gospodarowanie pamięcią ułatwiają instrukcje:

MEM — wyznacza liczbę wolnych bajtów w pamięci

FRE — wyznacza wielkość obszaru przeznaczonego na łańcuchy

VARPTR — określa adres zmiennej w pamięci.

Instrukcje **PRINT i PRINT TAB** są uzupełnione przez:

POS — badanie aktualnej pozycji kursora

PRINT @ — wpisanie znaku na wskazaną pozycję ekranu; ekran składa się z 1024 komórek utworzonych przez 16 linii po 64 znaki

PRINTUSING — drukowanie tekstu według określonego wzorca.

Dostępne są następujące instrukcje obsługi ekranu:

CLS — zerowanie całego ekranu

POINT (x,y) — testowanie punktu graficznego o współrzędnych x,y

SET (x,y) — rozjaśnienie punktu ekranu o współrzędnych x,y

RESET (x,y) — zaciemnienie punktu graficznego o współrzędnych x,y gdzie $0 \leqslant x \leqslant 127$ i $0 \leqslant y \leqslant 47$.

Do wywoływania podprogramów w języku maszynowym służy instrukcja USR (x), gdzie x jest parametrem z zakresu —32768 do +32767. Wynik działania procedury może być przekazany do programu w języku BASIC. Programy w języku maszynowym również mogą być ładowane z magnetofonu i uruchamiane do wskazanego adresu.

Komunikacja MERITUM z otoczeniem jest ułatwiona dzięki instrukcjom: INP (wczytuje bajt danych ze wskazanego portu) oraz OUT (wysyła dane do podanego portu).

Kilkanaście poleceń operatorskich umożliwia poprawianie programu. Jednak proces redagowania może odbywać się tylko w zakresie jednej linii programu. Duże znaczenie praktyczne mają polecenia TRON i TROFF, pozwalające na śledzenie przebiegu uruchamianego programu.

WSTĘPNA OCENA MIKROKOMPUTERA

O wartości mikrokomputera decyduje realnie dostępne oprogramowanie oraz jakość dokumentacji systemu. Pierwsze uwagi użytkowników MERITUM dotyczą znacznego nagrzewania się mikrokomputera i zasilacza, zwłaszcza podczas dłuższej pracy. Spodziewamy się, że może to być przyczyną różnych awarii i zakłóceń w pracy.

Dokumentacja techniczno-ruchowa mikrokomputera (31 stron formatu A-5) jest źle opracowana. Jej treść pozwala jedynie na bardzo powierzchowne zapoznanie się z urządzeniem. Rysunki i schematy włączono do instrukcji serwisowej, a producent od razu zastrzega, że nie będzie ona udostępniana posiadaczom MERITUM-I.

Ani opis sprzętu, ani opis oprogramowania nie pozwalają na zapoznanie się z programowaniem w języku wewnętrznym. Informacje o możliwościach dostępu do zasobów sprzętowych (np. układów we-wy) ograniczają się do krótkiej tabeli podającej adresy portów. Użytkownik odsyłany jest do katalogów firm zachodnich. Znaleźliśmy też szereg niejasności czy wręcz sprzeczności w szczegółach dokumentacji:

— na str. 5 podano, że interfejs magnetofonu jest asynchroniczny, a na str. 12 pokazano sygnał zapisu z bitami synchronizacji,

— na str. 21 opis wtyku do podłączenia monitora nie jest zgodny ani z gniazdem w monitorze, ani z tym co dostaje odbiorca,

(W szczególności nie wyprowadzono sygnału fonii.)

— na str. 5 wymieniono „repertuar" znaków semigraficznych — 96, repertuar znaków graficznych — 128". Podręcznik oprogramowania na str. 89 podaje tylko 64 znaki graficzne,

— nie podano opisu sygnałów na złączu interfejsu równoległego.

Dokumentacja oprogramowania to opis języka BASIC-MERITUM na 89 stronach A-5 z niewielką liczbą przykładów. Jest on wydrukowany starannie — lecz niestety, jego przejrzystość i komunikatywność nie jest zadowalająca. Opis języka jest bardzo powierzchowny i nie zawiera wielu istotnych informacji — choćby liczby wymiarów tablic, zakresu indeksów w tablicach itd. Bardzo przydałby się opis pamięci ROM — np. adresy początkowe i charakterystyka procedur oraz bliższe dane o zmiennych i tablicach systemowych itp. Nie dostarczono nam kasety z kursem programowania, mimo że jest ona przewidziana do kompletu urządzenia. Nie wiadomo dziś, czy będzie dostępne jakiekolwiek dodatkowe oprogramowanie — choćby w BASICU.

ZBIGNIEW CHRZEŚCIJAŃSKI
JACEK ŻEBROWSKI
Łódź

LITERATURA

[1] Dokumentacja techniczno-ruchowa mikrokomputera MERITUM-I nr ewid. 44990000

[2] Podręcznik programowania i użytkowania BASIC-MERITUM nr ewid. 44990001

[3] Korga Z., Smoliński A., Lipowski J., Podsiadło P.: Komputer osobisty MERITUM. Biuletyn Techniczno-Informacyjny MERA, nr 10/1983, s. 3

[4] Korga Z.: Mikrokomputer osobisty — MERITUM. Biuletyn Techniczno-Informacyjny MERA, nr 5—6/1984, s. 52.

Wracamy do początkowo zakładanej koncepcji kącika dla miłośników FORTHA. Pauza, choć nie do końca zaplanowana przez redakcję, umożliwiła przekazanie czytelnikom zasobu „szkolnych" informacji poprzez obszerne publikacje w części artykułowej INFORMATYKI. Dzięki temu możemy teraz odstąpić od opisywania podstaw FORTHA i zająć się publikowaniem ciekawych procedur (czyli tzw. słów). W poniższym tekście nawiązujemy do opisu implementacji FIG-FORTHA opisując słowo pozwalające na wprowadzenie pewnej użytecznej modyfikacji.

Zapraszamy amatorów FORTHA do nadsyłania opisów własnych słów, które mogą okazać się przydatne dla innych Czytelników.

FORTH (5)

PROSTE PROGRAMY NARZĘDZIOWE

LOOK-IN — „podglądanie" zawartości słowa zapisanego w Słowniku (dla słów definiowanych w FORTH bez ASSEMBLERA).

Słowa pomocnicze:
(wszystkie słowa definiowane przy użyciu HEX)

FIND — odszukanie CFA dla słowa podanego w łańcuchu wejściowym. Słowo to występuje w standardzie FORTH 79 lecz brak go w niektórych implementacjach. Na przykład w FIG-FORTH (CP/M) występuje —**FIND.**

```
: FIND  ( DLA FIG-FORTH )
        -FIND
        IF DROP 2 -
        ELSE ." BRAK SLOWA"
             BELL        ( : BELL 7 EMIT ; )
        ENDIF ;

: FIND-PFA      ( JAK FIND LECZ PFA )
        FIND 2 + ;

: NFA-FOR-CFA   ( ODSZUKANIE NFA PRZY ZNANYM CFA )
        3 - BEGIN 1 - C@ 80 > UNTIL ;

: NAME? ( CZY ADRES SLOWA ZDEFINIOWANEGO W FORTHIE )
        DUP NFA-FOR-CFA DUP C@
        80 - + 3 + = ;

: NAME(CFA)     ( PODANIE NAZWY SLOWA O ZNANYM CFA )
        NFA-FOR-CFA
        ID.     ( IDDOT WYSWIETLA SLOWO ) ;

: NAME(PFA)     ( PODANIE NAZWY SLOWA O ZNANYM PFA )
        2   NAME(CFA) ;

: L1    ( SLOWO CZESCIOWE DLA LOOK-IN, PODAJE ROZWIJANE SLOWO )
        FIND-PFA CR DUP ." : " NAME(PFA) CR ;

: L2    ( SLOWO CZESCIOWE DLA LOOK-IN )
        (PODAJE NAZWE KOLEJNEGO SKLADNIKA )
        DUP @ OVER ." ADR=" . ." HEX=" DUP . DUP
        ." NAME=" NAME? IF NAME(CFA) ELSE ." ????" DROP ENDIF ;

: ;S?   ( SLOWO CZESCIOWE DLA LOOK-IN )
        ( SPRAWDZA CZY KONIEC DEFINICJI )
        2 + DUP @ 2 +     ( NASTEPNY ADRES )
        ' ;S = ;          ( ' ODKLADA PFA SLOWA ;S )
                          ( KONCZACEGO DEFINICJE )
```

```
: (.")  ( CZY NA STOSIE JEST CFA SLOWA ." ? )
        DUP @ 2 +
        ' (."). ( ' DOKLADA PFA DLA NASTEPNEGO SLOWA )
        = ;

: "?    ODSZUKUJE KONIEC NAPISU DLA ." )
        DUP 2 + @ FF AND        ( OBLICZENIE DLUGOSCI )
        SWAP OVER 1 + OVER + ROT ROT 2 + ;

: E00FF ( WYSWIETLENIE ZNAKU Z LEWEGO BAJTU )
        DUP @ FF AND EMIT ;

: EFF00 (WYSWIETLENIE ZNAKU Z PRAWEGO BAJTU )
        DUP @ FF00 AND 100 / EMIT ;


: E."   ( WYSWIETLENIE NAPISU )
        SWAP 1 - SWAP EFF00 2 + ( PIERWSZY ZNAK )
        OVER IF BEGIN E00FF OVER 1 -    ( LEWY BAJT )
                IF EFF00 ENDIF  ( PRAWY BAJT )
                2 + SWAP 2 - SWAP OVER
                1 < UNTIL
             ENDIF DROP DROP ;


: LOOK-IN       ( PODGLADANIE ZAWARTOSCI ZDEFINIOWANEGO SLOWA )
        BASE @  ( ZAPAMIETANIE AKTUALNEGO SYSTEMU )
        HEX     ( USTALENIE SYSTEMU SZESNASTKOWEGO )
        L1      ( NAZWA ROZBIERANEGO SLOWA )
        BEGIN
                L2      ( NAZWA KOLEJNEGO SKLADNIKA )
                (.")?   ( CZY NAPIS ? )
                IF "? E." ENDIF CR      ( WYSWIETLENIE NAPISU )
                ;S?     ( CZY KONIEC DEFINICJI )
        UNTIL
        ." ; " CR DROP  ( KONIEC SLOWA )
        BASE !  ( POWROT DO STAREGO SYSTEMU )
```

PRZYKŁAD ZASTOSOWANIA LOOK-IN

System operacyjny CP/M staje się w kraju standardem dla mikrokomputerów 8-bitowych. Coraz popularniejszy staje się jednocześnie FIG-FORTH, implementacja FORTHA pod kontrolą systemu CP/M.

Przykład zastosowania procedury LOOK-IN jest jednocześnie receptą na drobną modyfikację FIG-FORTHA. Wersja standardowa zakłada, że konsola komunikuje się z systemem słowami o długości 7 bitów. Procedura **ID. (iddot)** przesyłająca nazwę słowa według zadanego NFA (adresu pola nazwy), nie obcina najbardziej znaczącego bitu w ostatnim znaku nazwy zdając się na odpowiednią obsługę sprzętową. Z tego powodu w nowszych konstrukcjach mikrokomputerów, gdzie układ wyświetlający dysponuje możliwościami graficznymi i wykorzystuje pełny bajt przy komunikacji z CPU — procedura **ID.**, a także **VLIST** (oparta na ID.) nie działa poprawnie.

Aby usunąć tą niedogodność wystarczy w **iddot** zastąpić **TYPE** przez własną procedurę TYPE-ASC.

```
: TYPE-ASC      ( zerowanie najbardziej znaczącego bitu i TYPE )
        OVER OVER  ( powielenie parametrów TYPE )
        0 DO
           DUP@ 7F7F AND  ( pobranie, odrzucenie bitu )
           OVER ! 1+      ( i zapisanie ponownie )
        LOOP DROP
        TYPE  ( wyświetlenie )
```

Przykład - modyfikacja FIG-FORTH (podane adresy są przykładowe)

1. Znalezienie adresu (CFA) dla TYPE-ASC:

```
FIND TYPE-ASC . ( cr ) 267F OK
```

2. Odszukanie TYPE w ID. :

```
1 3 ! ( przełączenie systemu na drukarkę wg CP/M )
LOOK-IN ID. ( cr )
```

(spodziewany wydruk):

```
: ID.
adr=E4A   hex=CD2    name=PAD
adr=E4C   hex=156    name=LIT
adr=E4E   hex=20     name=????
adr=E50   hex=156    name=LIT
adr=E52   hex=5F     name=????
adr=E54   hex=C7A    name=FILL
adr=E56   hex=556    name=DUP
adr=E58   hex=950    name=PFA
adr=E5A   hex=91C    name=LFA
adr=E5C   hex=52D    name=OVER
adr=E5E   hex=847    name=-
adr=E60   hex=CD2    name=PAD
adr=E62   hex=549    name=SWAP
adr=E64   hex=31D    name=CMOVE
adr=E66   hex=CD2    name=PAD
adr=E68   hex=AE7    name=COUNT
adr=E5A   hex=156    name=LIT
adr=E6C   hex=1F     name=????
adr=E6E   hex=3C3    name=AND
adr=E70   hex=AFA    name=TYPE
adr=E72   hex=8C1    name=SPACE
;
OK
```

Wracamy do klawiatury:

```
0 3 ! (cr) OK
```

i wpisujemy CFA dla TYPE-ASC w miejsce CFA TYPE:

```
267F E70 ! (cr) OK
```

Teraz już tylko dla sprawdzenia:

```
VLIST (cr)
```

MAREK CZARZYŃSKI
Warszawa

Żyrardowskie Zakłady Tkanin Technicznych

96-300 ŻYRARDÓW, ul. Okrzei 51

oferują do sprzedaży:

- pamięć półprzewodnikową do minikomputera MERA 400 o pojemności 32 K słów
- urządzenia do obsługi taśmy papierowej:
  - czytnik CT 2100 M1
  - perforator DT 105 S
  - rozwijacz RT 2200
  - zwijacz ZT 2200
- dziurkarki kart Soemtron 415–1
- sprawdzarkę kart Soemtron 425–1

Informacji udziela Dział Informatyki, tel. 20-31 do 35 w. 284

EO/142/K/85

Wbrew wszelkim poczynaniom, które „nie pomagają" w rozwoju mikroinformatyki, coraz więcej osób zaczyna się w Polsce posługiwać mikrokomputerami. Najbardziej popularnego dla mikrokomputerów — języka BASIC — można się nauczyć z instrukcji obsługi. Mówiąc precyzyjnej, nowicjusz może dowiedzieć się z instrukcji obsługi jakie jest działanie poszczególnych instrukcji i — zacząć programować. Aby jednak programować dobrze, trzeba mieć za sobą lata doświadczeń lub... skorzystać z doświadczeń innych.

W Akademii mikroKLANU nie piszemy jak programować, lecz — jak lepiej programować. Wychodzimy przy tym z założenia, że dobrze napisany program zawsze można kosztem odstępstwa od publikowanych reguł dopasować jakoś do mniejszych zasobów pamięci w komputerze. Natomiast źle napisany program, choćby zajmował minimalną ilość miejsca w pamięci, szybko idzie do lamusa.

Bardziej doświadczonych programistów zapraszamy do przedstawienia własnych refleksji.

Akademia MikroKLANU (2)

Silnia

W pierwszym odcinku omawiane było wyjście z programu. Przede wszystkim apelowaliśmy o pełne wydruki. Mamy jednak nadzieję, że Czytelnicy nie potraktowali naszych rad dosłownie — spotyka się wydruki składające się z kilkudziesięciu stron, gęsto zapisane liczbami. Takie wyjście jest gorsze niż w przypadku zbyt skąpej informacji: użytkownik programu sam musi spełniać rolę komputera i wyszukiwać potrzebne mu liczby — niczym igłę w stogu. Nieco przewrotnie przedstawiamy więc program ilustrujący poprawne wejście, który drukuje bardzo dużo cyfr — ale takie jest jego przeznaczenie!

Zasady organizacji wejścia do programu są proste: należy przedrukować wszystko, co zostało wprowadzone do komputera. Jest to najlepszy sposób kontroli poprawności wprowadzania danych, szczególnie jeżeli program — tak dzieje się w naszym przykładzie — długo działa. Zasadą o mniejszym znaczeniu jest reguła automatycznego wprowadzania wszystkich stałych oraz typowych wyborów. Dotyczy to w szczególności wyboru drukarki lub monitora ekranowego jako urządzenia wyjścia. Program, który korzysta z obu możliwości, powinien zapytać użytkownika o rodzaj wyjścia, sugerując bardziej prawdopodobne lub wygodniejsze urządzenie. W naszym przykładzie nie mamy takich problemów. Warto natomiast zwrócić uwagę, że program sugeruje jaka może być największa liczba, dla której obliczenia mogą być jeszcze wykonane przy danym wymiarze wektora A. Taka konstrukcja programu (mimo uwag wstępnych, patrz linie 5 oraz 6) pozwala go wykorzystywać również tym, którzy nie znają treści programu (użytkownik wcale nie musi umieć programować). Zasada, którą tu zastosowaliśmy, opiera się na zwielokrotnieniu informacji ważnych z punktu widzenia operacyjnego.

Ważną sprawą, która pojawia się podczas analizy przykładowego programu, jest także optymalizacja obliczeń. Szczegółowe uwagi o takiej optymalizacji — zarówno z punktu widzenia wykorzystania pamięci komputera, jak i szybkości obliczeń — można znaleźć w świetnej książce D. van Tassela „Praktyka programowania", WNT, 1978. W naszym programie wewnętrzna pętla (po J) będzie powtarzana wielokrotnie razy (dla dużych N). Każde uproszczenie programu w tej pętli pozwoli skrócić (długi!) czas jego realizacji. Dlatego też, mimo mniejszej czytelności tak zaprogramowanego wyrażenia, zdecydowaliśmy się zastąpić dzielenie przez dziesięć mnożeniem przez 0.1. Zamiana ta rzeczywiście redukuje czas obliczeń, gdyż dla każdego komputera dzielenie jest operacją dłuższą niż mnożenie. Podobne reguły dla innych działań można znaleźć we wspomnianej książce Tassela.

A oto nasz program szkolny II — służy on do obliczania silni dużych liczb z pełną dokładnością, tzn. podaje wszystkie cyfry. Zwróćmy uwagę, że standardowy BASIC nie pozwala wyprowadzać informacji w instrukcji INPUT — dlatego też najpierw drukujemy pytanie o liczbę, której silnię będziemy obliczali, a dopiero potem występuje instrukcja wejścia. Dla większości znanych mikrokomputerów instrukcje te można połączyć w jedną:

```
20 INPUT "LICZBA (<1001)";N
```

Jednak pisząc program dla wszystkich musimy trzymać się reguł składni języka BASIC. Trochę to niewygodne, ale dzięki temu nasz program może być uruchomiony na każdym komputerze wyposażonym w interpreter lub kompilator BASICA!

```
5 REM OBLICZANIE SILNI BARDZO DUZYCH LICZB
6 REM (N<1001) Z PELNA DOKLADNOSCIA.
10 DIM A(3000)
20 PRINT "LICZBA (<1001)?"
21 INPUT N
30 PRINT N;"!=";
40 D=1
50 C=0
60 A(1)=1
70 FOR I=2 TO N
80 FOR J=1 TO D
90 A(J)=A(J)*I+C
100 C=INT(0.1*A(J))
110 A(J)=A(J)-10*C
120 NEXT J
130 IF C=0 THEN GOTO 190
140 C1=INT(0.1*C)
150 D=D+1
160 A(D)=C-10*C1
170 C=C1
180 GOTO 130
190 NEXT I
200 FOR I=D TO 1 STEP -1
210 PRINT A(I);
220 NEXT I
230 END
READY.
    RUN
LICZBA (<1001)?
? 15
 15 != 1 3 0 7 6 7 4 3 6 8 0 0 0
READY.
```

Zwróćmy jeszcze uwagę, że dla zupełnie głupiego użytkownika (tacy też się, niestety, zdarzają) można by dodać linię 25:

```
25 IF N <1 OR N> 1001 THEN GOTO 20
```

która radykalnie zabezpieczy przed niewłaściwym wejściem. Ponieważ nasz program ma bardzo długi czas realizacji, więc wykrycie, że wprowadzona liczba N jest większa niż 1001 w pierwotnej wersji programu nastąpi dopiero pod koniec obliczeń! Czy zastosujemy tak radykalne zabezpieczenie w programie, który piszemy tylko dla siebie? Decyzja powinna zależeć przede wszystkim od grona potencjalnych użytkowników — tak naprawdę nigdy nie piszemy żadnego programu tylko dla siebie. Łatwo przekonać się o tym, gdy będziemy dysponowali dobrymi programami — na pewno wielu znajomych będzie chciało się z nami wymieniać.

Życząc Czytelnikom szerokiego rozpowszechnienia ich wspaniałych programów kończymy posiedzenie Akademii.

JAKUB TATARKIEWICZ

Ośrodek Obliczeniowy PPiMUEB „Elektromontaż"

70-370 Szczecin, ul. Bohaterów Warszawy 15/16
tel. 450-02 w. 26 lub 78, teleks 0422210

zakupi:

dwa minikomputery MERA 100

w dowolnej konfiguracji

EO/138/K/85